# ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

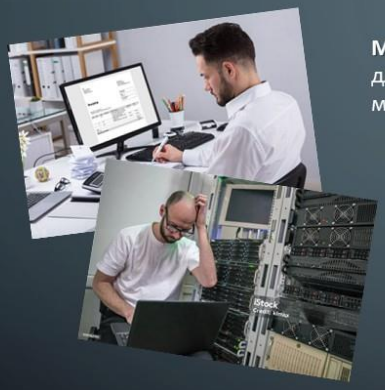
Кваліфікаційна робота
Перший (бакалаврський) рівень

# ВЕБ-ПЛАТФОРМА ДЛЯ МЕРЕЖІ БУДІВЕЛЬНИХ МАГАЗИНІВ

ВИКОНАВ:
ЗДОБУВАЧ. ГР. КІУКІУ-22-2
ЦАРЕНКО Д.О.

КЕРІВНИК:
ДОЦЕНТ КАФЕДРИ ЕОМ
ФІЛІМОНЧУК Т.В.

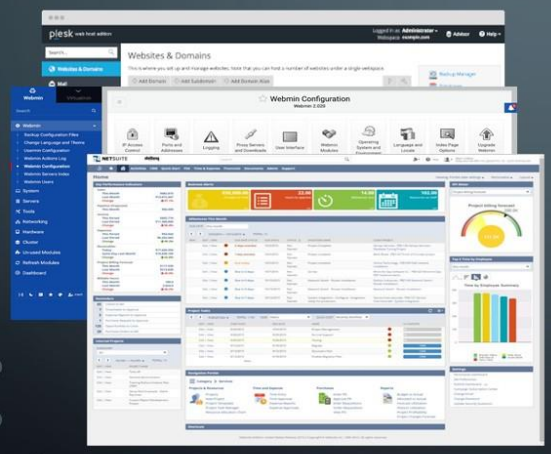
# МЕТА ТА ЗАДАЧІ РОБОТИ




**Мета:** створення комплексного програмного забезпечення (ПЗ) для автоматизації процесу продажу та керування ресурсами мережі будівельних магазинів

**Задачі:**
- аналіз ринку, переваг та недоліків продуктів-конкурентів;
- планування веб-інфраструктури проєкту;
- вибір технологій, інструментів та підходів до взаємодії компонентів веб-застосунку;
- кодування, тестування та відлагодження усіх рівнів ПЗ;
- забезпечення додаткових умов роботи ПЗ.

# ІСНУЮЧІ АНАЛОГІ

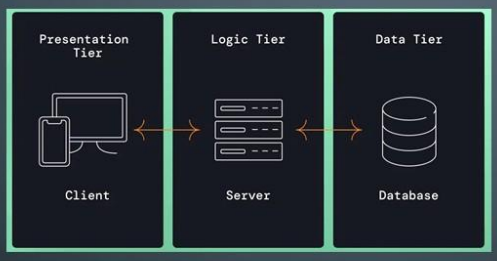



**Найчастіші недоліки:**
- важка крива навчання для новачків;
- висока вартість використання;
- відсутня сумісність для певних ОС або пристроїв;
- обмеження на кастомізацію.

**Найчастіші переваги:**
- простота у використанні;
- масштабованість;
- можливість глобального використання;
- широке ком'юніті.

3

# ПІДХІД ДО РЕАЛІЗАЦІЇ




Під час розробки було створено програмне забезпечення на основі тришарової архітектури з «тонким» клієнтом

Протокол передачі даних – HTTP,
метод – POST,
формат – JSON.

4

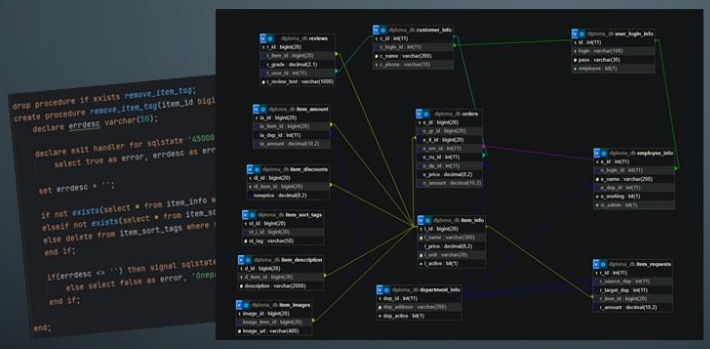
# ВИКОРИСТАНІ ТЕХНОЛОГІЇ

**Фронтенд:**

**Бекенд:**

5

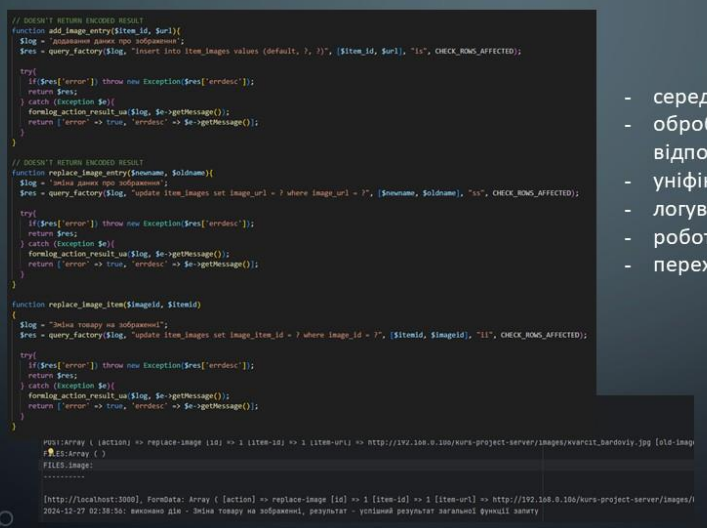
# БАЗА ДАНИХ



- Зберігає дані додатку
- Зберігає процедури для спрощення багатокрокових дій
- Використовується для створення зв'язків між таблицями даних
- Широко застосовує транзакції
- Процедури мають уніфікований формат результату

6

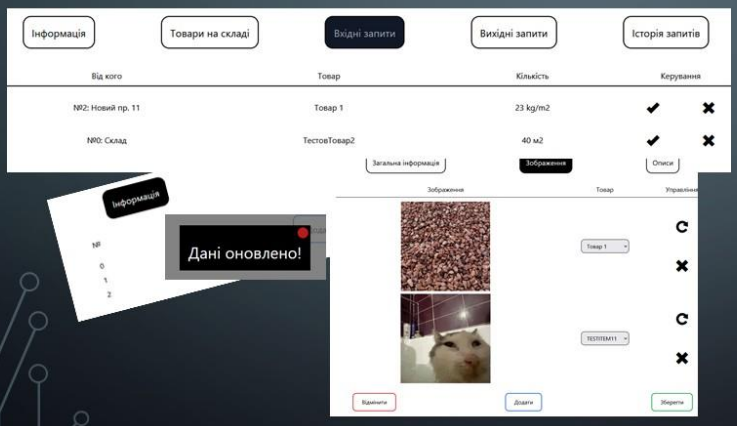
# СЕРВЕРНИЙ ЗАСТОСУНОК



- середня ланка між клієнтом та БД;
- оброблює передані клієнтом дані, повертає відповідь;
- уніфікований набір функцій;
- логування запитів;
- робота з файлами;
- перехоплювач POST-запитівю

7

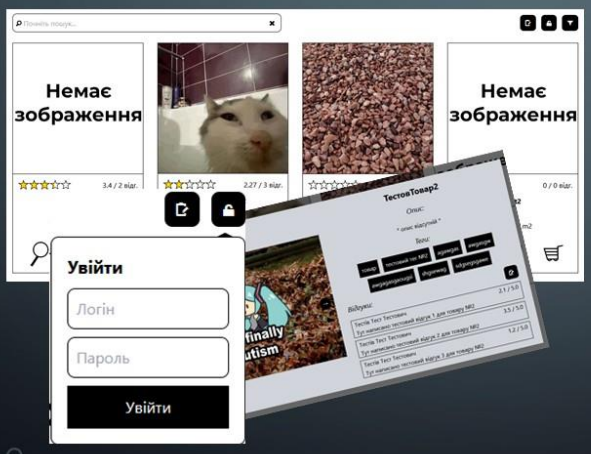
# АДМІНІСТРАТИВНИЙ ЗАСТОСУНОК




- графічний інтерфейс для внесення змін у БД;
- два рівні привілеїв;
- призначений для використання працівниками мережі;
- перевикористовує шаблонні компоненти сторінки;
- основа роботи – fetch().

8

# КОРИСТУВАЦЬКИЙ ЗАСТОСУНОК




- веб-магазин;
- показує дані з БД у відповідному форматі;
- може змінювати таблиці відгуків на товар, замовлень та облікові дані одного користувача.

9

# ВИСНОВКИ

- створено тришаровий веб-застосунок, кожен шар має чітко відведену роль та не створює конфліктів логіки;
- реалізована робота з БД, в тому числі санітизація запитів та даних;
- реалізовано три можливі ролі користувача застосунку, графічний інтерфейс змінюється відповідно ним;
- модульний підхід до розробки дозволяє адаптувати та масштабувати створене ПЗ до нових умов.

10

# ДОДАТОК Б

## Код створення бази даних

```
drop database if exists diploma_db;
create database diploma_db;
use diploma_db;
drop table if exists user_login_info;
create table user_login_info(
    id int unique primary key auto_increment,
    login varchar(100) unique not null,
    pass varchar(30) not null,
    employee bit
) charset utf8;
drop table if exists department_info;
create table department_info(
    dep_id int primary key unique auto_increment,
    dep_address varchar(200),
    dep_active bit
) charset utf8;
drop table if exists customer_info;
create table customer_info(
    c_id int unique auto_increment primary key,
    c_login_id int,
    c_name varchar(200) not null,
    c_phone varchar(10) unique not null,
    foreign key (`c_login_id`) references `user_login_info`
(`id`)
) charset utf8;
drop table if exists employee_info;
create table employee_info(
    e_id int unique primary key auto_increment,
    e_login_id int,
    e_name varchar(200) not null,
    e_dep_id int,
    e_working bit,
    is_admin bit,
    foreign key (`e_login_id`) references `user_login_info`
(`id`),
    foreign key (`e_dep_id`) references `department_info`
(`dep_id`)
) charset utf8;
drop table if exists item_info;
create table item_info(
    i_id bigint unique auto_increment primary key,
    i_name varchar(300) not null,
    i_price decimal(8,2) not null,
    i_unit varchar(20),
    i_active bit
) charset utf8;
```

```
drop table if exists item_images;
create table item_images(
    image_id bigint unique primary key auto_increment,
    image_item_id bigint,
    image_url varchar(400),
    foreign key (`image_item_id`) references item_info (`i_id`)
) charset utf8;
drop table if exists item_amount;
create table item_amount(
    ia_id bigint unique auto_increment primary key,
    ia_item_id bigint,
    ia_dep_id int,
    ia_amount decimal(10, 2) check (ia_amount > 0.0),
    foreign key (`ia_item_id`) references item_info (`i_id`),
    foreign key (`ia_dep_id`) references department_info
(`dep_id`)
) charset utf8;
drop table if exists reviews;
create table reviews(
    r_id bigint primary key auto_increment,
    r_item_id bigint,
    r_grade decimal(2, 1) check ( r_grade >= 0.0 and r_grade <=
5.0 ),
    r_user_id int,
    r_review_text varchar(1000),
    foreign key (r_item_id) references `item_info` (`i_id`),
    foreign key (r_user_id) references `customer_info` (`c_id`)
) charset utf8;
drop table if exists item_requests;
create table item_requests(
    r_id int primary key auto_increment,
    r_source_dep int,
    r_target_dep int,
    r_item_id bigint,
    r_amount decimal(10,2),
    foreign key (`r_target_dep`) references department_info
(`dep_id`),
    foreign key (`r_source_dep`) references department_info
(`dep_id`),
    foreign key (`r_item_id`) references item_info (`i_id`)
) charset utf8;
drop table if exists item_description;
create table item_description(
    d_id bigint primary key auto_increment,
    d_item_id bigint unique,
    description varchar(2000),
    foreign key (`d_item_id`) references `item_info`(`i_id`)
) charset utf8;
drop table if exists item_discounts;
create table item_discounts(
    di_id bigint primary key auto_increment,
    di_item_id bigint unique,
```

```
    newprice decimal (8,2),
    foreign key (`di_item_id`) references `item_info`(`i_id`)
) charset utf8;
drop table if exists item_sort_tags;
create table item_sort_tags(
    st_id bigint primary key auto_increment,
    st_i_id bigint,
    st_tag varchar(50)
) charset utf8;
drop table if exists orders;
create table orders(
    o_id bigint primary key auto_increment,
    o_gr_id bigint not null,
    o_it_id bigint not null,
    o_em_id int not null,
    o_cu_id int not null,
    o_dp_id int not null,
    o_active bit,
    o_date varchar(20),
    o_price decimal(8,2) not null check ( o_price > 0.0 ),
    o_amount decimal(10,2) not null check ( o_amount > 0.0 ),
    foreign key (`o_dp_id`) references `department_info`
(`dep_id`),
    foreign key (`o_it_id`) references `item_info` (`i_id`),
    foreign key (`o_em_id`) references `employee_info`(`e_id`),
    foreign key (`o_cu_id`) references `customer_info`(`c_id`)
) charset utf8;
drop procedure if exists check_login;
create procedure check_login(_login varchar(30), _pass
varchar(100))
begin
    declare error bit;
    declare is_employee bit;
    declare hits smallint;

    set hits = (select count(id) from user_login_info where
login = _login and pass = _pass);

    if(hits = 1) then begin
        set error = 0; set is_employee = (select employee from
user_login_info where login = _login and pass = _pass limit 1);
        end;
    else begin
        set error = 1; set is_employee = 0;
        end;
    end if;

    select error as error, is_employee as employed;
end;
drop procedure if exists  get_user_info;
create procedure get_user_info(_login varchar(30), _pass
varchar(100))
begin
```

```
    if ((select count(id) from user_login_info where login =
_login and pass = _pass) = 0) then select true as error,
'користувача не знайдено' as errdesc;
    else begin
        declare id_hit int; declare employee_dep int; declare
employee_admin bit;
        declare employed bit;
        set id_hit = (select id from user_login_info where pass
= _pass and login = _login limit 1);
        set employed = (select employee from user_login_info
where id = id_hit);
        if(employed) then
            set employee_dep = (select e_dep_id from
employee_info where e_login_id = id_hit limit 1);
            set employee_admin = (select is_admin from
employee_info where e_login_id = id_hit limit 1);

            if((select e_working from employee_info where
e_login_id = id_hit) = 0) then select true as error, 'працівник
тимчасово недоступний' as errdesc;
            else select false as error, true as emp, (select
e_name from employee_info where e_login_id = id_hit) as name,
(select dep_address from department_info where dep_id = (select
e_dep_id from employee_info where e_login_id = id_hit)) as
address,
                        employee_dep as dep_id, employee_admin
as is_admin, id_hit as id;
            end if;
        else select false as error, false as emp, (select c_name
from customer_info where c_login_id = id_hit) as name, (select
c_phone from customer_info where c_login_id = id_hit) as phone;
        end if;
    end;
    end if;
end;
drop procedure if exists add_employee;
create procedure add_employee(tempinfo varchar(25), temppass
varchar(20), name varchar(200), dep_id int, working bit, admin
bit) begin
    declare newlogin_id int;
    declare temp_login varchar(30);
    set temp_login = concat('user_', tempinfo);
    if (select count(id) from user_login_info where login =
temp_login) = 1 then select true as error, 'Такі дані входу вже
існують' as errdesc ;
    else begin
        insert into user_login_info values (default, temp_login,
temppass, true);
        set newlogin_id = (select id from user_login_info order
by id desc limit 1);
        insert into employee_info values (default, newlogin_id,
name, dep_id, working, admin);
```

```
        if(select count(e_id) from employee_info where e_name =
name and e_dep_id = dep_id and e_working = working and is_admin
= admin and
            e_login_id = (select id from user_login_info where
login = temp_login and pass = tempinfo)) is null then select
true as error, 'Дані не додано' as errdesc;
        else select false as error, temp_login as login,
temppass as password;
        end if;
    end;
    end if;
end;
drop procedure if exists form_item_request;
create procedure form_item_request(source int, target int, item
bigint, amount decimal(10,2)) begin
    declare error bit; declare errdesc varchar(300);
    if(select dep_id from department_info where dep_id = source)
is null then begin
        set error = true; set errdesc = 'Відділення-створювач
запиту з таким ID не існує';
    end;    else
    if(select dep_id from department_info where dep_id = target)
is null then begin
        set error = true; set errdesc = 'Відділення-отримувач
запиту з таким ID не існує';
    end;    else
    if source = target then begin
        set error = true; set errdesc = 'Не можна створити запит
на це ж відділення';
    end;    else
    if(select i_id from item_info where i_id = item) is null
then begin
        set error = true; set errdesc = 'Такого товару не
існує';
    end;    else
    if(select ia_item_id from item_amount where ia_dep_id =
target) is null then begin
        set error = true; set errdesc = 'Товару на складі-
отримувачі немає';
    end;    else
    if(select ia_amount from item_amount where ia_dep_id =
target and ia_item_id = item) < amount then begin
        set error = true; set errdesc = 'Кількість запрошеного
товару перевищує кількість на складі отримувача запиту';
    end;
    end if; end if; end if; end if; end if; end if;
    if (error = true) then select error as error, errdesc as
errdesc;
    else begin
        insert into item_requests values (default, source,
target, item, amount);
        select false as error, 'Успішно' as errdesc;
    end;
```

```
    end if;
end;
drop procedure if exists confirm_incoming;
create procedure confirm_incoming(id bigint) begin
    declare error bit;
    declare errdesc varchar(100);
    declare amount_s decimal(10,2);
    declare needed decimal(10,2);
    declare amount_t decimal(10,2);
    declare exit handler for sqlstate '45000' begin
        rollback ;
        select true as error, errdesc as errdesc;
    end;
    set amount_t = (
        select ia_amount from item_amount where
            ia_dep_id = (select r_target_dep from item_requests
where r_id = id) and ia_item_id = (select r_item_id from
item_requests where r_id = id)
        );
    set amount_s = (
        select ia_amount from item_amount where
            ia_dep_id = (select r_source_dep from item_requests
where r_id = id) and ia_item_id = (select r_item_id from
item_requests where r_id = id)
        );
    set needed = (select r_amount from item_requests where r_id
= id);
    if needed > amount_t then begin set error = true; set
errdesc = 'На складі немає необхідної кількості товару'; end;
    else begin
        start transaction;
        update item_amount set ia_amount = (ia_amount - needed)
where ia_dep_id = (select r_target_dep from item_requests where
r_id = id);
        if(select row_count()) = 0 then begin set errdesc =
'Віднімання даних від отримувача провалено'; signal sqlstate
'45000'; end; end if;
        update item_amount set ia_amount = (ia_amount + needed)
where ia_dep_id = (select r_source_dep from item_requests where
r_id = id);
        if(select row_count()) = 0 then begin set errdesc =
'Додавання даних до надсилача провалено'; signal sqlstate
'45000'; end; end if;
        delete from item_requests where r_id = id;
        if(select row_count()) = 0 then begin set errdesc =
'Видалення запису про запит провалено'; signal sqlstate '45000';
end; end if;
        commit;
        set error = false; set errdesc = 'Успішно';
    end;
    end if;
    select error as error, errdesc as errdesc;
end;
```

```
drop procedure if exists add_new_tag;
create procedure add_new_tag(item_id bigint, tag varchar(50))
begin
    declare errdesc varchar(50);
    declare exit handler for sqlstate '45000' begin
        select true as error, errdesc as errdesc;
    end;
    set errdesc = '';
    if not exists(select i_id from item_info where i_id =
item_id) then set errdesc = 'Не знайдено товару з таким
ідентифікатором!';
    elseif exists(select * from item_sort_tags where st_i_id =
item_id and st_tag = tag) then set errdesc = 'Такий теґ на цьому
товарі вже існує!';
    else insert into item_sort_tags values (default, item_id,
tag);
    end if;
    if(errdesc <> '') then signal sqlstate '45000';
        else select false as error, 'Операція успішна' as
errdesc;
    end if;
end;
drop procedure if exists remove_item_tag;
create procedure remove_item_tag(item_id bigint, tag
varchar(50)) begin
    declare errdesc varchar(50);
    declare exit handler for sqlstate '45000'
        select true as error, errdesc as errdesc;
    set errdesc = '';
    if not exists(select * from item_info where i_id = item_id)
then set errdesc = 'Не знайдено товару з таким
ідентифікатором!';
    elseif not exists(select * from item_sort_tags where st_i_id
= item_id and st_tag = tag) then set errdesc = 'Такого теґу на
заданому товарі не додано!';
    else delete from item_sort_tags where st_i_id = item_id and
st_tag = tag;
    end if;
    if(errdesc <> '') then signal sqlstate '45000';
        else select false as error, 'Операція успішна' as
errdesc;
    end if;
end;
drop procedure if exists register_customer;
create procedure register_customer (_name varchar(200), _phone
varchar(10), _login varchar(100), _password varchar(30)) begin
    declare errdesc varchar(60);
    declare customer_login_id bigint;
    declare exit handler for sqlexception begin
        rollback ;
        select true as error, 'Внутрішня помилка бази даних при
додаванні користувача' as errdesc;
    end;
```

```
    if(select id from user_login_info where login = _login limit
1) is not null then
        select true as error, 'Логін вже існує' as errdesc;
        else begin
            start transaction ;
            insert into user_login_info values (default, _login,
_password, false);
            set customer_login_id = (select id from
user_login_info where login = _login and pass = _password limit
1);
            insert into customer_info values (default,
customer_login_id, _name, _phone);
            if(select id from user_login_info where id =
customer_login_id limit 1) is null or
              (select c_id from customer_info where c_login_id =
(select id from user_login_info where login = _login and pass =
_password)) is null then begin
                rollback ;
                select true as error, 'Дані не було коректно
додано!' as errdesc;
                  end;
            else
            commit ;
            end if;
            select false as error, customer_login_id as errdesc;
        end;
    end if;
end;
```